Japanese Patent Application KOKAI No. 58-215795

Date of KOKAI: December 15, 1983

Title of the Invention: Non-volatile memory device

Application No. 57-98308

filed June 8, 1982

Inventor: Yoshiyuki Tanaka

Applicants: Tokyo Shibaura Denki K.K.

Specification

1. Title of the Invention:

Non-volatile memory device

2. The Claim:

A device consisting of an electrically programable non-volatile memory, wherein exclusive location used as a counting area which stores the number of times data was written there is partly allocated.

3. Detailed Explanation of the Invention

(Field of the invention)

This invention relates to a non-volatile memory device, and especially to the electrically programable semiconductor non-volatile memory device.

(Background of the invention)

The semiconductor non-volatile memory device consists of MOS-FET's for storing binary data. Each MOS-FET stores charge as binary data, and it keeps charge stored although no power is

supplied to the memory cells.

Various types of non-volatile memories have been proposed, and UV-EPROM (ultraviolet erasable and programable ROM) is one of the most popular memories of these types. If UV-EPROM is exposed to the ultraviolet UV rays outside the circuit, charges are erased from or written into the UV-EPROM.

An EEPROM(electrically erasable and programable ROM) is another one of the most popular memories of these types. Data can be erased from or written into the EEPROM while the EEPROM is being installed in the circuit if the erasing-and-writing device is used together with the EEPROM circuit. An EEPROM can thus be used for such a system that the stored contents may change frequently, typically a cash register.

An EEPROM can be used together with a static RAM so as to constitute a non-volatile RAM. The EEPROM in the non-volatile RAM provides the same capacity as the static RAM, and it can operate in the same mode while the power is on. When the power goes off, information stored in the EEPROM is kept stored although information in the static RAM disappears. At that time, control goes to the EEPROM starting with the static RAM. When the power goes on again, information moves from the EEPROM to the static RAM. The above processes are the way of keeping the information stored while the power is off.

A high voltage is required for writing data into an EEPROM, and this high voltage limits the alteration of memory

contents and also the number of times data was reprogramed. The number of times data was reprogramed is currently limited to 1,000 to 10,000 for an EEPROM. This limitation is mandatory, and the reliability is not assured for the reprograming exceeding the limited number of times.

The principle of operation of the EEPROM and the limitation on the number of times data was reprogramed will be described hereafter. Figure 1 shows the cross-sectional view of the typical EEPROM cell. Figure 1(a) shows the charge flow when data is written, and Figure 1(b) shows the charge flow when data is erased. In Figure 1, first polycrystalline silicon layer forming first electrode 11, second polycrystalline silicon layer forming floating gate 12, and third polycrystalline silicon layer forming second erase-and-write electrode 13 are provided on P-type silicon substrate 10 together with insulation layer 14 made of SiO_2. Floating gate 12 is located between first and second electrodes 11 and 13, and floating from these electrodes.

For the programing of data, first electrode 11 is set at a voltage of 0V or common potential, and second electrode 13 is set at high voltage (+V). (See Figure 1(a).) At that time, floating gate 12 goes to high voltage (+V) due to static coupling to second electrode 13. This causes a high electric field between first electrode 11 and floating gate 12. Thus, electrons move from first electrode 11 to floating gate 12 due

to tunneling and these electrons are captured by floating gate 12. After electrons are completely captured by floating gate 12, second electrode 13 is set at a voltage of 0V to complete the programing. At that time, floating gate 12 is kept at negative voltage since electrons are being captured by floating gate 12.

Next, consider that the corresponding cell has been programed by the electrons captured by floating gate 12. (See Figure 1(b).) For the erase of data, first electrode 11 is set at a voltage of 0V, floating gate 12 at a voltage of 0V, and second electrode 13 at high voltage (+V). A high electric field is generated between floating gate 12 and second electrode 13, and electrons move from floating gate 12 to second electrode 13 through insulation layer 14 due to tunneling. When electrons captured by floating gate 12 are completely moved to second electrode 13, the erasing of data is completed. At that time, second electrode 13 is set at a voltage of 0V.

As described above, the programed state is defined as that in which electrons have been captured by floating gate 12 to such a level that in which floating gate 12 is set at negative voltage. The erased state is defined as that in which electrons captured by floating gate 12 have completely been moved away so that floating gate 12 is set at a voltage of 0V. The former corresponds to logical 1 and the latter to logical

0, and vice verse. The programed state becomes logical 1 in some cases, and logical 0 in the other case. Logical states are defined by peripheral devices.

The number of times data is reprogramed in an EEPROM is limited due to the motion of electrons from first electrode 11 to floating gate 12 by tunneling when second electrode 13 is set at high voltage for the programing of data. That is: electrons move from first electrode 11 to floating gate 12 through the insulation layer of SiO_2 and vice versa. This type of electron motion puts stress on the insulation layer and it degrades the insulation layer. The insulation layer is not so strongly stressed when data is written into the cell while data is being erased, and the degree of degradation is quite few.

When programs are frequently altered in the EEPROM installed in a certain system, the memory contents may disappear in some cases as described above. The maximum number of times data will be reprogramed in an EEPROM is determined in accordance with the expected life of the EEPROM in a certain system, and the EEPROM is thus replaced in predetermined time intervals. This maintainance method is doubtful from the view point of reliability.

(Objective of the invention)

The objective of the present invention is to provide the non-volatile memory device wherein the number of times data is reprogramed can be assured to assure high reliability.

(Outline of the invention)

The non-volatile memory device built in accordance with the present invention provides a special memory cell location wherein the number of times data was reprogramed is stored so that one could know the remaining number of times data can be reprogramed.

(Advantages of the invention over the technique in the prior art)

In accordance with the present invention, one can easily know the remaining number of times data can be reprogramed in the corresponding memory device, and the memory contents never disappear in such a system that data is frequently reprogramed. Thus, the reliability can be improved in such a manner as described above. Unlike the decision on the EEPROM device life in the prior art, the number of times data was reprogramed in the device can accurately be known. Thus, the device is never replaced before its life, and the device can be used economically.

(Embodiments of the invention)

The present invention will be described hereafter referring to the drawings.

Figure 2 shows the memory areas of an EEPROM built in accordance with the present invention.

In Figure 2, 1 indicates the entire memory areas, and 2 indicates the exclusive location called the counter area

wherein the number of times data was reprogramed is stored. Memory area 1 includes counter area 2. The number of bits in counter area 2 is determined by the total number of times data can be reprogramed. Counter area 2 thus occupies the required number of bits in memory area 1.

How to count the number of times data was reprogramed will be described hereafter.

First, counter area 2 is initialized (typically to 0). The contents of counter area 2 are read out of the memory device each time a program is written into the memory area, and at that time the number which is read out of counter area 2 is incremented (typically by one). The updated number is then stored in counter area 2 again. Next, a program is written into the memory area other than counter area 2.

Although the contents of counter area 2 are first incremented and thereafter a program is written into the memory area other than counter area 2, a program can first be written into the memory area other than counter area 2 and thereafter the contents of counter area 2 can be incremented. The process of this operation depends on the design criteria.

Figure 3 shows the memory areas of the non-volatile RAM consisting of an EEPROM and an RAM. In Figure 3, 3 indicates the RAM area, and the other numbers for the EEPROM are as indicated in Figure 2. Counter area 2 is first initialized (typically to 0). Data is normally read from or written into

RAM 3 during the system operation. The contents of RAM 3 are transferred to the EEPROM to keep them stored when the power is off in the system.

Before the contents of RAM 3 are transferred to the EEPROM, data is read out of exclusive location 4 (counter area 4) which corresponds to exclusive location 2 in the EEPROM. Data read out of counter area 4 is incremented and then written into counter area 4 again. After the contents of counter area 4 are updated in this manner, the contents of RAM 3 are stored in memory area 1. At that time, the contents of counter area 4 are stored in counter area 2.

Before RAM 3 is used again, the contents of memory area 1 and counter area 2 are respectively written (recalled) into RAM 3 and counter area 4.

The number of times data was reprogramed is thus recorded and kept stored as part of the non-volatile memory contents. This informs the user of the remaining number of times data can be reprogramed, and this number never disappear.

[Variation of the invention]

In the above embodiment, the contents of counter area 2 or 4 are incremented every time data is reprogramed. If the maximum permissible number of times for the reprograming is preset as initial data, the contents of the counter area can be decremented every time data is reprogramed. At that time, the remaining number of times data can be reprogramed can be known.

When the number of times data was reprogramed becomes a predetermined value, a display typically a CRT display or a lamp indicator can inform the operator of the excess number of times or the prohibition of reprograming.

(Brief description of the drawings)

Figure 1 shows the cross-sectional views of an EEPROM, where data has been written into the cell in Figure 1(a), and data has been erased in Figure 1(b).

Figure 2 shows an embodiment of the memory device built in accordance with the present invention.

Figure 3 shows another embodiment of the memory device built in accordance with the present invention.

1...memory area

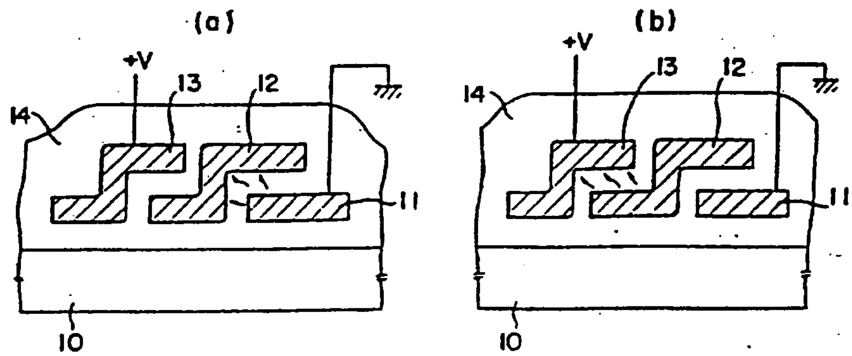
2...counter area

3...RAM

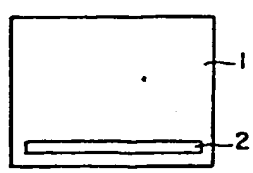
4...counter area

514

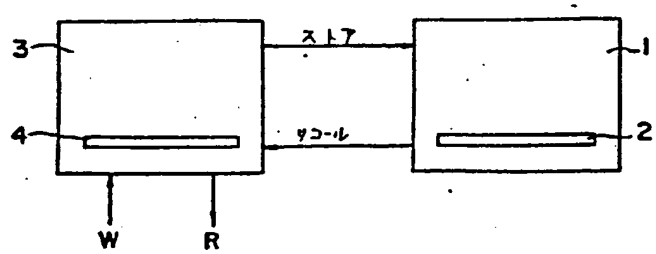
第 1 図



第 2 図



第 3 図



Japanese Patent Application KOKAI No. 62-283496

Date of KOKAI: December 9, 1987

Title of the Invention: Write operation of a programable read only memory

Application No. 61-124731

filed May 31, 1986

Inventor: Shinichi Nakada

Applicants: Canon INC.

Specification

1. Title of the Invention:

Write operation of a programable read only memory

2. The Claim:

System of managing the write operations to the programable read only memory wherein data written in the memory area thereof can electrically be erasable; where

said memory area is divided into a plurality of blocks,

the number of times data was written into each block is stored,

data is written from blocks where the frequency of times data is written is low into unused blocks in order in the ascending order of the frequency of times data is written into the former blocks,

said blocks where the frequency of times data is written is low are connected behind said unused blocks.

3. Detailed Explanation of the Invention:

(Field of the invention)

This invention relates to the management of the number of time data can repetitively be written into an electrically erasable, programable read only memory.

(Background of the invention)

An EEPROM (electrically erasable and programable ROM) in the early time was small in capacity and operated with a great amount of peripheral circuits for a write operation. The peripheral circuits in the early time were operated in such an erase mode that all data stored in the EEPROM chip is erased at a time. Recently, EEPROM is increased in capacity and operated with fewer peripheral circuits even if it is connected to the address and data buses of the CPU. Any one byte of data can be erased from EEPROM right now. Due to the improvement described above, the random access memory (RAM) in the prior art can be replaced by the EEPROM for special purpose.

The memory card is used to keep the alphabetical characters, programs, and/or sentences stored after they are generated from a machine, typically a small personal computer or a word processor for Japanese characters. Since the memory card consists of a battery and an RAM, it can receive the alphabetical characters, programs, and/or sentences from the machine, when set in the machine, and also keep them stored after pulled out of the machine. If the memory card is built with an EEPROM, the battery may be omitted.

(Objective of the invention)

The number of times data can repetitively be written into the EEPROM is however limited unlike the RAM since there are no limitation on the RAM. This implies that data can disappear when written into the EEPROM beyond the limited number of times data can repetitively be written into the memory card consisting of an EEPROM. Some data is frequently replaced by new one, and other data is rarely replaced by new one. If the number of times the former is rewritten exceeds the limit on

the EEPROM, the contents of the EEPROM cannot be replaced any more even though the number of times the latter is rewritten into the EEPROM is within the limitation.

Since the present invention is disclosed to solve the above problem, the objective of the present invention is to provide the system of managing the number of times data can repetitively be written into the programable read only memory, whereby the life of the EEPROM can be extended through the protection of written data against disappearing, averaging of the number of times of write operations for every EEPROM cell, and averaging of the frequency of times of write operations for every EEPROM cell.

(Summary of the invention)

The present invention describes the system of managing the number of times data can repetitivery be written into the programable read only memory, which is characterized in that the memory area is divided into a plurality of blocks, the number of times data was written into each block, data of the respective blocks where the frequency of times data is written thereinto is low is written into unused blocks starting from the beginning thereof in an ascending order of said number of times data was written into each block, and said blocks where the frequency of times data is written thereinto is low are connected behind the unused blocks.

(Principle of operation)

In the present invention, the number of times data was written into each block of the memory area is stored, data of the blocks where the frequency of times data was written into each block is low is written into the unused blocks in an ascending order of the number of times data was written into

each block, and the blocks where the frequency of times data was written into each block is low are connected behind the unused blocks.

(Embodiments of the invention)

Figure 1(a) shows the pointer and its spare blocks which are used to manage the number of times data can be written into the programable read only memory in accordance with the present invention. In Figure 1(a), 1 indicates the EEPROM with a capacity of 32788 bytes by 8 bits, whereinto data can be written up to 10,000 times. EEPROM 1 consists of pointer block 1a and its spare blocks SPB1 through SPB50. Pointer block 1a contains 4 addresses, each consisting of one byte. Addresses "0" and "1" constitute a write counter (WCNT) consisting of 2 bytes and they can store the number of times data was repetitively written. They typically indicate "1388_{16}". Address "2" of pointer block 1a, consisting of one byte, stores directory DB, typically "01_{16}". Address "3" of pointer block 1a, consisting of one byte, stores unused start block address OSB, typically "33_{16}". Address "4" of pointer block 1a, consisting of one byte, stores unused end block address OEB, typically "08_{16}".

Figure 1(b) shows the block diagram of the system configuration as an embodiment in accordance with the present invention. In Figure 1(b), 11, 11a, and 11b indicate the CPU, ROM, and RAM, respectively. The system operation is controlled by the program specified by the flowcharts of Figures 7 and 8 which are being stored in ROM 11a. In addition, 12 indicates input means which specify the operations to write data into EEPROM 1 which has been installed in write data device 13 and to erase data from EEPROM 1. CPU 11 contains accumulators ACC

and BCC which perform the executions, respectively.

Figure 2 shows the configuration of a file in EEPROM 1 of Figure 1(a). In Figure 2, 21 indicates block addresses, typically BLOCK 1 through BLOCK 127. Each block typically consists of 256 bytes, where the leading 2 bytes store the number of times data was updated in the corresponding block. The succeeding 253 bytes keep data stored. The ending one byte thereof stores continue block area CB which indicates whether data being stored is complete or is continued to any other block. If data continues to any other block, continue block area CB stores the block address to which data continues. Unless data continues to any other block, continue block area CB stores FF_{16}.

Figure 3 shows the configuration of the directory block of Figure 2. In Figure 3, 30 indicates the directory block specified by said directory DB, 31 indicates the update counter to update the contents of said directory block 30, which typically consists of 2 bytes, 32 indicates the file area which stores the filename, typically consisting of 12 bytes, 33 indicates the start block address area (SB) which stores the start block address of the corresponding file, typically consisting of one byte, 34 indicates the end block address area (EB) which stores the end block address of the corresponding file, typically consisting of one bytes, and 35 indicates the chaining block area (CB) which stores information of whether any directory block follows directory block 30 or not, typically "FF_{16}". Directory block 30 typically consists of 15 file areas 32.

The structure of EEPROM 1 will be described hereafter referring to Figures 1(a) and 3.

If write counter WCNT in bytes 0 and 1 of pointer block 1a indicates typically "1388_{16}" as shown in Figure 1(a), they mean that data was updated 5,000 times. In Figure 1(a), directory DB stores "01_{16}". That is, the block address of directory block 30 specified by directory DB is "1", and update counter 31 of directory block 30 stores "$142F_{16}$". The contents of update counter 31 indicate that the contents of directory block 30 were updated 5167 times. In file 1 (FiLe) (filename) of file area 32, start block address area 33 is specified as "02_{16}" and end block address area 34 as "05_{16}". They mean that file 1 starts at block BLOCK 2 and terminates at block BLOCK 5. In file 2 of file area 32, start block address area 33 is specified as "$0A_{16}$" and end block address area 34 as "$0F_{16}$". They mean that file 2 starts at BLOCK 10 and terminates at block BLOCK 15. In file 3 (filename) of file area 32, start block address area 33 is specified as "15_{16}" and end block address area 34 as "18_{16}". They mean that file 3 starts at BLOCK 21 and terminates at BLOCK 24. Continue block area following file 3 of file area 33 is specified as "FF_{16}" and it means that file area 33 ends with file 3.

Figure 4 shows the configuration of the unused blocks in EEPROM 1, wherein the elements having appeared in Figures 1(a) and 3 are assigned by the same numbers.

For the unused blocks of EEPROM 1 in Figure 4, write counter WCNT of pointer block 1 is specified as "0001_{16}", directory DB thereof as "01_{16}", unused start block address OSB thereof as "02_{16}", and unused end block address OEB thereof as "$7A_{16}$". These bytes occupy addresses 0 through 4 of pointer block 1a. In block BLOCK 1 specified by directory DB, update counter 31 is specified as "0001_{16}", file 1 of file area 32 as

"FF_{16}", and chaining block area 35 as "FF_{16}". They indicate that EEPROM 1 is unused.

Start block address OSB of pointer block 1a is specified as "02_{16}" and end block address OEB thereof as "$7E_{16}$". Leading 2 bytes of blocks BLOCK 2 through BLOCK 127 are respectively specified as "0001_{16}". Chaining block areas 35 for blocks BLOCK 2 through BLOCK 126 are respectively specified as "03_{16}" through "$7E_{16}$", which indicate the continuation of blocks, and chaining block area 35 for block BLOCK 127 as "FF_{16}". Note that the chaining block area is the one byte located at the end of each block. As described above, blocks BLOCK 2 through BLOCK 127 are chained.

How to write data into EEPROM 1 will be described hereafter referring to Figures 3, 5(a), and 5(b).

Figures 5(a) and 5(b) show the pointer and directory blocks used for writing data into EEPROM 1, respectively. In Figures 5(a) and 5(b), the elements having appeared in Figures 1(a) and 3 are assigned by the same number. Assume that the memory contents are as shown in Figure 3 right before data is written into EEPROM 1.

Control finds "00_{16}" from leading bytes of file area 32 in a certain block BLOCK#. In Figure 3, "00_{16}" are located between files 2 and 3. At that time, control writes filename "file 4" there in 12 bytes. Referring to start block address OSB of an unused block in pointer block 1a, control increments the contents of update counter 31 by one, or increments data in leading 2 bytes of block BLOCK 57_{16} specified by start block address OSB. If the contents of update counter 31 exceed typically 10,000, control performs the same operations as above for the block BLOCK# specified by chaining block area 35 of

file 4 until control finds a block BLOCK# whose update counter 31 indicates a number of less than 10,000. At that time, control writes the block address of that block into start block address area OSB of pointer block 1a and also writes data of file 4 into block BLOCK 87 (253 bytes). If block BLOCK 87 overflows, control increments by one the contents of update counter 31 for block BLOCK# specified by chaining block area 35 of block BLOCK 87. Thereafter, control checks if the contents of that update counter 31 exceed typically 10,000. If update counter 31 indicates a number of greater than 10,000, control finds a block BLOCK# whose update counter 31 indicates a number of less than 10,000, and control writes the block address of that block BLOCK# into chaining block area 36 of block BLOCK# whereinto data was written right before. Data is written into blocks in this manner so that blocks BLOCK# whereinto data was repetitively written 10,000 times or more can be rejected. The same operation is repeated until no data is written into EEPROM 1. When data is written into a last block, control writes new unused start block address OSB in place of the contents of chaining block area 35 of said last block BLOCK#. Thereafter control increments by one the contents of write counter WCNT of pointer block 1a so that said write counter WCNT indicates "1389_{16}", and enters "FF_{16}" into chaining block area 35 of said last block BLOCK#. Tnereafter, control writes the address of said last block BLOCK# into end block address area 34 which stores the last block address of directory block 30, and increments by one the contents of update counter 31. The contents of update counter 31 then becomes "1430_{16}" as shown in Figure 5(b). At that time, start block address area 33 of file 4 is specified as "33_{16}" and end block address area 34 thereof

as "37_{16}".

How to delete file 1 from EEPROM 1 will be described hereafter referring to Figures 5(a) and 5(b).

Control finds file 1 from block Block 1 since block BLOCK 1 is used as directory block 30, and then control specifies leading 2 bytes of file area 32 as "00_{16}". Control then increments by one the contents of update counter 31 of directory block 30. Referring to the contents of both start block address area 33 and end block address area 34 of file 1, control changes the contents (which were "FF_{16}" right before the deletion of file 1) of chaining block area 35 of the block specified by end block address OEB of pointer block 1a into the contents of start block address area 33. Control then increments by one the contents of update counter 31 of that block. This operation implies that control connects file 4 (which has been deleted) behind unused blocks. In this manner, files are repetitively updated and deleted while the contents of update counter 31 are incremented, and finally the contents of update counter 31 approach 10,000.

How to access EEPROM 1 when the contents of update counter 31 approach 10,000 will be described hereafter.

Control first sets new start block address OSB at chaining block area 35 of the block BLOCK# specified by start block address OSB of pointer block 1a. Control then transfers any data other than the contents of update counter 31 of directory block 30 located right before said block specified by start block address OSB of pointer block 1a. Thereafter, control writes new directory block address into directory DB of pointer block 1a, increments by one the contents of write counter WCNT of pointer block 1a, and also increments by one the contents of

update counter 31 thereof.

If the contents of write counter WCNT of pointer block 1a exceed 10,000, control transfers any other data than the contents of write counter WCNT to one of spare pointer blocks SPB1 through SPB50 in the nearest location, and increments by one the contents ("0000_{16}") of write counter WCNT of new pointer block so that the contents of write counter WCNT become "0001_{16}". At that time, the contents of write counter WCNT of pointer block 1a, which have been disregarded, become greater than 10,000, and the contents of write counter WCNT of new pointer block 1a become less than 10,000. The delete and write operations for both directory block 30 and pointer block 1a are managed in this manner. Control moves the block used by deleted life behind the last unused block so that the frequency of times for the use of unused blocks can be averaged. Unless a used file is updated, the number of times data was updated in the block used by that file does not change. If any file remains cataloged without its use after created, the number of times data was updated in that file is in some cases much less than the number of times data was updated in the other block. For instance, the former is 2 while the latter is 5,000 or more. The number of times data was written into each block of EEPROM 1 is thus to be averaged.

The complementary processing for the averaging can start under condition (a) or (b) described below.

(a). The contents of write counter WCNT of pointer block 1a become a multiple of 256.

(b). The difference between the lowest average value for the contents of update counter 31 of the blocks, each of which constitutes a file, and the average value for the contents of

update counter 31 of the unused blocks exceeds 256.

After creating a new file or deleting a file, control checks if the contents of write counter WCNT of pointer block 1a become a multiple of 256 or if the low order byte of loading 2 bytes becomes "00_{16}". At the time the above condition becomes valid, control performs the test for the files in accordance with the order that the files have been cataloged into directory block 30. Control accumulates the contents of update counters 31 for the blocks which constitute a file, and divides the sum of these contents by the number of blocks which constitute that file so that the average number of times data was updated could be obtained. Thereafter, control calculates the average number of times data was updated in each of the other files. Control then compares the averages among the others so that the lowest number of times data was updated could be used as a reference for the comparison. Control then finds the file whose number of times data was updated is the lowest. Control calculates the average number of times data was updated in unused blocks, and also calculates the difference between this average and the reference obtained before.

The complementary processing will be described referring to Figures 6(a) through 6(c).

Figures 6(a) through 6(c) show the configuration of blocks when the complementary processing is carried out for the averaging. In Figures 6(a) through 6(c), 41 indicates the file consisting of blocks BLOCK 5 through BLOCK 7, and the average number of times data was updated therein is lowest. In addition, 42 indicates a group of unused blocks, which is specified by unused start block address OSB of pointer block

1a. Unused block group 42 consists of chaining blocks BLOCK 50, BLOCK 10, BLOCK 11, BLOCK 18, BLOCK 55, BLOCK 80, and BLOCK 81.

Control connects the start block of file 41 behind block BLOCK 81 of unused block group 42 in the following manner. That is, the contents of chaining block area 35 of block BLOCK 81 are changed from "FF_{16}", (shown in Figure 6(a)) to "05_{16}" (shown in Figure 6(b)) so that chaining block area 35 indicates block BLOCK 5 of file 41. Until the contents of chaining block area 35 of file 41 become "FF_{16}", control connects blocks BLOCK 5 through BLOCK 7 of file 41 behind block BLOCK 81 of unused block group 42. At that time, control specifies the start pointer for file 41 in directory block 30 as block BLOCK 50 and the end pointer therefor as block BLOCK 11, as shown in Figure 6(c). Thereafter, control specifies unused start block address OSB of unused block group 42 as "12_{16}" and unused end block address OEB thereof as "$0B_{16}$". All blocks used as part of a cataloged file can thus be used again, and blocks in EEPROM 1 can equally be used and updated.

Figure 7 shows the flowchart of the write operation for EEPROM of Figure 1(a). Items 1 through 18 indicate the processing steps.

Control finds an unused area from directory block 30, and writes a new filename there. (See step 1.) Control stores unused start block address OSB to accumulator ACC for CFULL. (See step 2.) Control increments by one the contents of write counter WCNT specified by accumulator ACC. (See step 3.) Control checks if the contents of write counter WCNT exceed 10,000. (See step 4.) If the response is YES, control stores continue block area CB of the block specified by accumulator

ACC in accumulator ACC and then returns to step 3. (See step 5.) If the response is No, control writes the contents of accumulator ACC into start block address area SB of directory block 30. (See step 6.) Thereafter, control writes data into the data area of the block specified by accumulator ACC. (See step 7.) Control then checks if data being written into the block specified by accumulator ACC exceeds the capacity of the block specified by accumulator ACC or 235 bytes. (See step 8.) If the response is YES, control stores continue block area CB of the block specified by accumulator ACC into accumulator BCC. (See step 9.) Next, control increments by one the contents of write counter WCNT of the block specified by accumulator BCC. (See step 10.) Thereafter, control checks if the contents of write counter WCNT exceeds 10,000. (See step 11.) If the response is YES, control stores continue block area CB of the block specified by accumulator BCC, and then returns to step 10. If the response is NO, control writes data of accumulator BCC into continue block area CB of the block specified by accumulator ACC, and returns to step 7.

If the response is NO at step 8, control writes continue block area CB specified by accumulator ACC into unused start block address OSB, (See step 14.) Next, control increments by one the contents of write counter WCNT of pointer block 1a. (See step 15.) Thereafter, control writes "FF_{16}" into continue block area CB specified by accumulator ACC. (See step 16.) Then, control writes data of accumulator ACC into end block address area 34 on a location allocated for a new file in directory block 30. (See step 17.) Control updates the contents of write counter WCNT of directory block 30. (See step 18.)

Figure 8 shows the flowchart of the complementary processing performed in accordance with the present invention, where items 1 through 7 indicate steps 1 through 7, respectively.

Control checks if the contents of write counter WCNT of pointer block 1a are a multiple of 256. (See step 1.) If the response is NO, control returns to the start. If the response is YES, control calculates the average for the contents of update counter 31 of the blocks constituting each file cataloged in directory block 30, and finds the file where the number of times data was updated is the lowest. (See step 2.) Control then calculates the average for the number of times data was updated in the unused blocks. (See step 3.) Thereafter, control subtracts the lowest value of the averages for the contents of update counters in the blocks which constitute the corresponding file from the average for the contents of the update counters of the unused blocks. Thereafter, control checks if the result of the subtraction is greater than 256. (See step 4.) If the response is NO, control returns to the start. If the response is YES, control connects the head of the corresponding file behind the unused blocks. (See step 5.) Thereafter, control transfers the contents of the file connected behind the unused blocks to one of the unused blocks. (See step 6.) Control alters the start and end pointers of the connected file in directory block 30. (See step 7.) Control then returns to the start.

(Advantages of the invention over the technology in the prior art)

In accordance with the present invention, as described above, the memory area is divided into a plurality of blocks,

the number of times data was written is stored for each block, data of blocks where the frequency of times data is written is low is written into the unused blocks starting with the beginning of the unused blocks in accordance with the ascending order of the number of times data was written, and the block where the frequency of times data can be rewritten is the lowest is connected behind the unused blocks. The above operations protects EEPROM against disappearing of data, and averages the number of times data was written into EEPROM. This invention averages the frequency of times data is written into each block of EEPROM, and the life determined by the number of times data can be written can be extended.

4. Brief description of the drawings

Figure 1(a) shows the pointer and its spare blocks used to manage the number of times data can be written into EEPROM in accordance with the present invention.

Figure 1(b) shows the block diagram of the system configuration as an embodiment in accordance with the present invention.

Figure 2 shows the configuration of a file in EEPROM.

Figure 3 shows the configuration of the directory block of Figure 2.

Figure 4 shows the configuration of the unused blocks in EEPROM.

Figures 5(a) and 5(b) show the pointer and directory blocks used for writing data into EEPROM, respectively.

Figures 6(a) through 6(c) show the configiration of blocks when the complementary operation is carried out for obtaining the average number of times data was written into blocks.

Figure 7 shows the flowchart of the write operation for

EEPROM 1 of Figure 1(a).

Figure 8 shows the flowchart of the complementary processing performed in accordance with the present invention.

1....EEPROM

1a...pointer block

21...block address

30...directory block

31...update counter

32...file area

33...start block address area

34...end block address area

35...chaining block area

41...file

42...unused block group

Figure 1(a)

1a: pointer block

SPB1: Spare Pointer Block

SPB2:

SPB49: Spare Pointer Block

SPB50: Spare Pointer Block

Figure 1(b)

12: Input Means

Figure 2

BLOCK 1 Stored Data

Figure 7

START

(1) WRITE NEW FILENAME.

(2) STORE UNUSED START BLOCK ADDRESS IN ACCUMULATOR.

(3) UPDATE NUMBER OF TIMES DATA WAS REPETITIVELY WRITTEN.

(4) NUMBER OF TIMES DATA WAS WRITTEN > 10,000?

(5) STORE CONTINUE BLOCK AREA IN ACCUMULATOR ACC.

(6) WRITE DATA OF ACCUMULATOR ACC INTO START BLOCK ADDRESS AREA.

(7) WRITE DATA INTO DATA AREA.

(8) CAPACITY OF BLOCK > 235 BYTES?

(9) STORE BLOCK AREA IN ACCUMULATOR.

(10) UPDATE NUMBER OF TIMES DATA WAS REPETITIVELY WRITTEN.

(11) NUMBER OF TIMES DATA WAS WRITTEN > 10,000?

(12) STORE CONTINUE BLOCK AREA IN ACCUMULATOR BCC.

(13) WRITE DATA OF ACCUMULATOR BCC INTO CONTINUE BLOCK AREA.

(14) WRITE DATA OF CONTINUE BLOCK AREA INTO UNUSED START BLOCK ADDRESS.

(15) UPDATE NUMBER OF TIMES POINTER BLOCK WAS REPETITIVELY WRITTEN

(16) WRITE "FF_{16}" INTO CONTINUE BLOCK AREA OF BLOCK SPECIFIED BY ACCUMULATOR ACC.

(17) WRITE DATA OF ACCUMULATOR ACC INTO END BLOCK ADDRESS AREA.

(18) UPDATE NUMBER OF TIMES DIRECTORY BLOCK WAS REPETITIVELY WRITTEN.

STOP

Figure 8

START

(1) NUMBER OF TIMES POINTER BLOCK WAS REPETITIVELY WRITTEN TO BE A MULTIPLE OF 256?

(2) FIND FILE WHERE NUMBER OF TIMES DATA WAS WRITTEN IS

LOWEST AFTER CALCULATING AVERAGE FOR CONTENTS OF UPDATE COUNTER.

(3) CALCULATE AVERAGE FOR NUMBER OF TIMES DATA WAS UPDATED IN UNUSED BLOCKS.

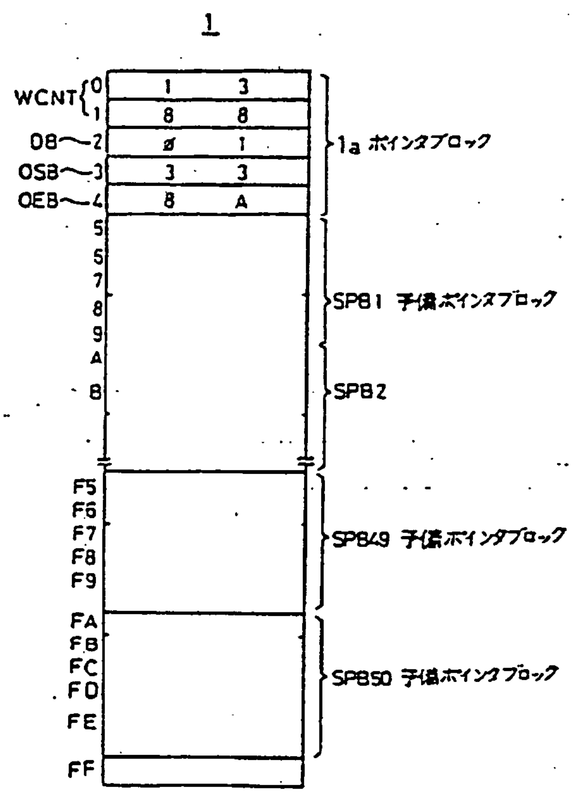
(4) SUBTRACTED VALUE -256 > 0?

(5) CONNECT HEAD OF CORRESPONDING FILE BEHIND UNUSED BLOCKS.

(6) TRANSFER CONTENTS OF CONNECTED FILE TO UNUSED BLOCK.

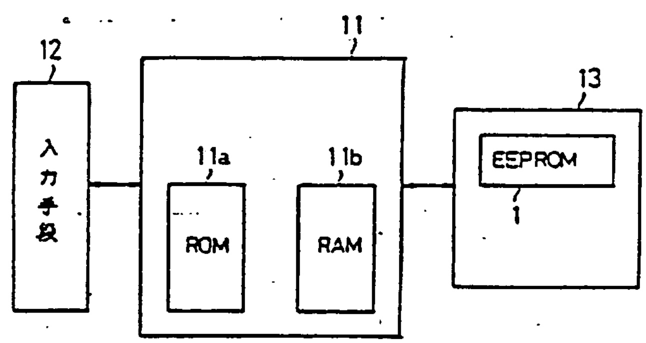
(7) ALTER START AND END POINTERS OF CONNECTED FILE.

第 1 図 (a)



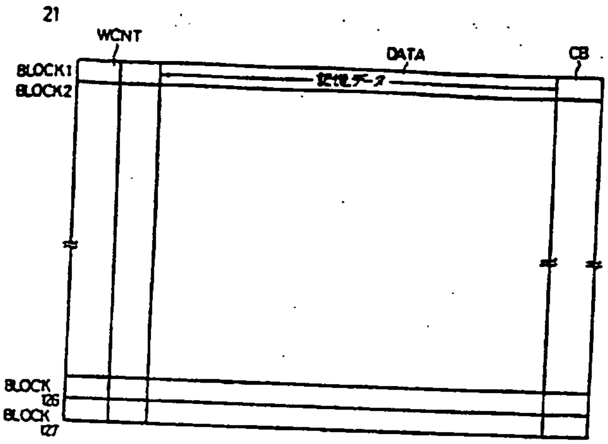
1
WCNT
0
1
1
3
8
8
DB〜2
ø
1
OSB〜3
3
3
OEB〜4
8
A
1a ポインタブロック
5
6
7
8
9
A
B
SPB1 予備ポインタブロック
SPB2
F5
F6
F7
F8
F9
SPB49 予備ポインタブロック
FA
FB
FC
FD
FE
SPB50 予備ポインタブロック
FF

第 1 図 (b)

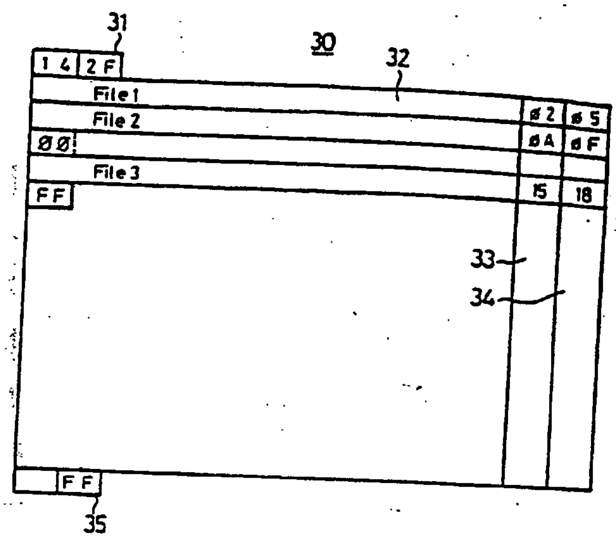


12
入力手段
11
11a
ROM
11b
RAM
13
EEPROM
1

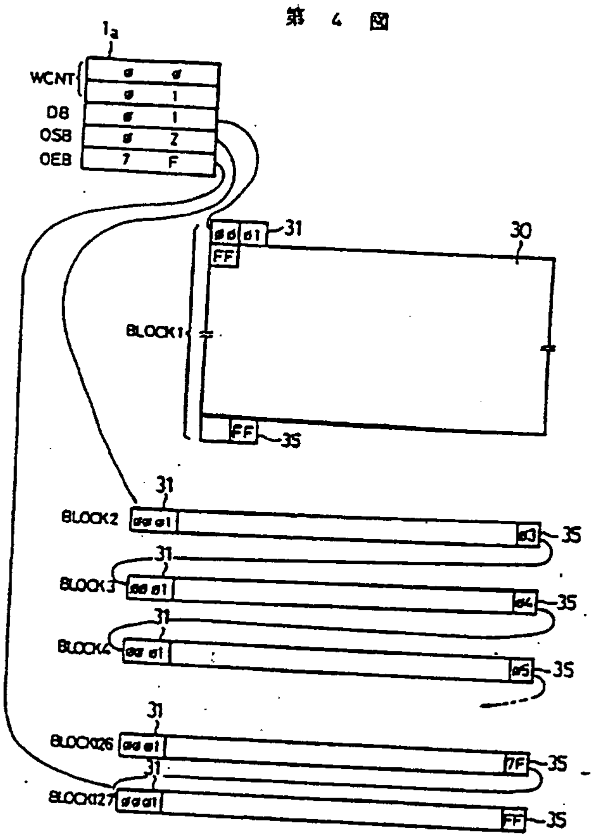
第 2 図



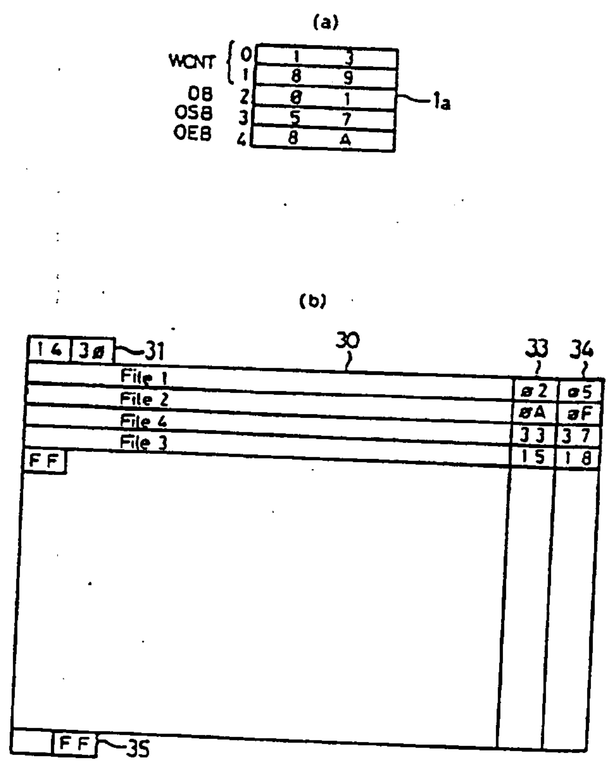
第 3 図



第 4 図

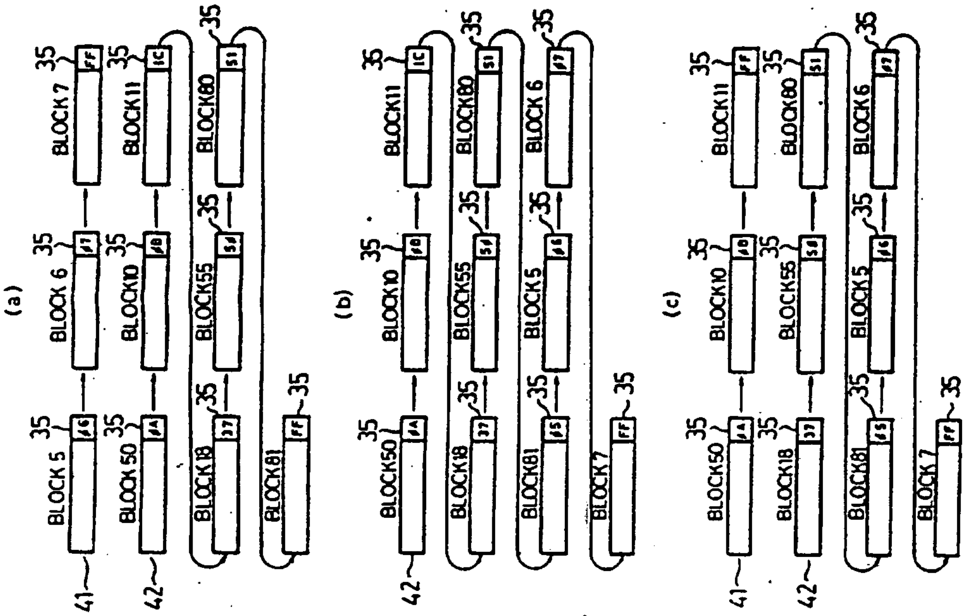


第 5 図

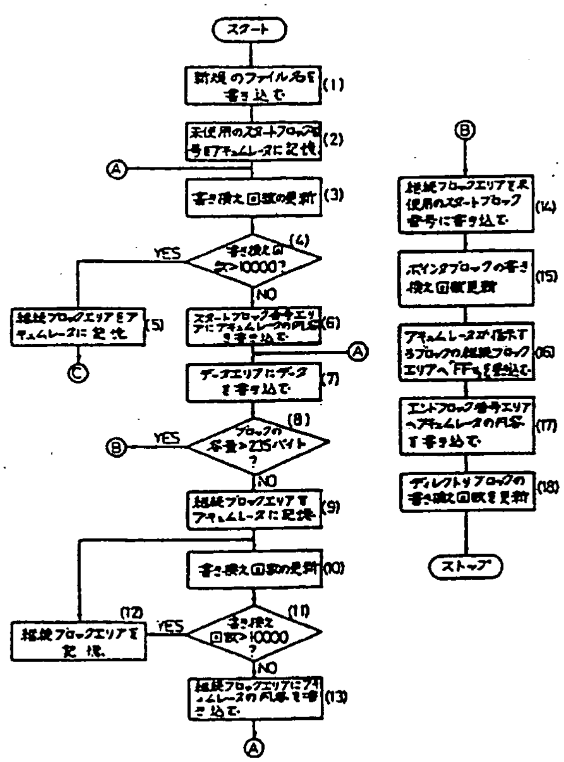


515

第 6 図



第 7 図



第 8 図